THE EVOLUTION OF A SCALABLE DEPARTMENTAL REAL-TIME CLINICAL INFORMATION SYSTEM

Andrew Galewsky, BS, Daniel Galewsky, MS, Perry Statham, Charles B. Owen, MD
Clinical Resource Systems, Inc., Austin, TX

Abstract

On-line Clinical Information Systems have to be both very responsive and capable of accessing large volumes of historic data. The EmStat Emergency Department Clinical Information System has evolved through several approaches to meeting these needs, including effective use of client/server technology, and optimizing all available machine resources. Currently the system is quite successful in this regard but the evolution continues.

A paradox in designing an on-line clinical information system is that not only do the users want the system to be very responsive, they want access to the volumes of historic data generated. The EmStat Emergency Department Information Systems has used several approaches since it first went on-line in 1986.

First we would like to present a little background on the system in general. It is a client/server system with the services residing on a UNIX system called the host. The graphical user interface resides on PC based workstations called CareStat. The services on the host include the database, text formatting, printing and communications.

The original design goal for the system was to maximize the use of limited hardware resources to try to optimize for speed through careful software design rather then relying on hardware. This includes extensive caching on both the host and workstation sides, careful tuning of the database queries and a simple yet robust network protocol based on UDP (user datagram protocol) packets. As hardware prices have fallen, allowing us to use more powerful machines these optimizations continue to add to the overall speed and scalability of the system. The system has so far scaled up to quite a large size for departmental systems, with the largest site running over 70 workstations and storing data for over 100,000 patient visits a year. This site wants to keep at least 5 years of information on-line and accessible for both research and past medical history.

The host for the first EmStat system was one of the first 386 motherboards running SCO XENIX (at the time) and an 80 Mb hard drive. The database software was a version of DB-Vista from the Raima Corporation. At that time the clinical workstations were color serial terminals or small combination touch-screen terminal/telephones and what where called PDS's (Personal Data Stations). These PDS's were small keypad/bar-code devices that were linked back to the host using RS-422 multidrop protocol. It was during the implementation of this configuration of the system that the problems of scalability were first encountered. We first designed the system to use a separate process for each terminal and PDS. As the system got out of the prototype stage it became obvious that the host resources were being consumed to great extent just trying to poll the devices (about 10 at the time), leaving little left over to do the other computing chores. Part of the difficulties were exacerbated by the fact that SCO XENIX had no mechanism to multiplex many inputs into one message stream. We decided that the next iteration of the system would be designed to allow one blocking input to relieve the polling burden on the CPU.

Shortly after the deployment of this system we began exploring the use of Microsoft Windows 2.0 for the graphic front end of the system. While the prototype revealed the benefits in relieving processing from the host computer, it also revealed the major shortcomings of Windows 2.0, especially as implemented for the 286 PC. It was at this stage we began to fully exploit the use of touch-screens as the primary data-entry mechanism. It had become obvious that requiring the clinicians to type data or even to use keyboards was not a viable option for the high level of compliance we expected. It seemed that the amount of training required for a keyboard based system was going to be excessive for the high turnover observed in the ED

(Emergency Department) staff. The touch-screen seemed to alleviate the problem. Everyone, it seemed, could be quickly taught how to touch a computer.

The database structure at this point in the development also began to reveal certain shortcomings.

The database was laid out in a real-time and archive set of schema. This would allow for the rapid response of the real-time minute to minute operation of the department and after the patient was discharged the data was moved to an archive portion of the database. While this provided for a high efficiency for the real-time portion of the system, it caused many headaches in creating reports. Many reports would be run while patients were currently in the department and spanning over into the archive portion of the database. Therefore every query had to take this into consideration causing the queries to become quite convoluted to ensure that patients from both portions of the database were correctly accounted for.

In addition the early versions of DB-Vista had only a network type of database and no SQL (Structured Query Language) based query language. It had no concept of UNION ALL or facilities for automatic replication of data. In addition the record structures had to be compiled into the programs, making it difficult to add columns to database records without recompiling all the programs. We also discovered that many of the recovery techniques for abnormal termination of the programs relied on reloading the database from the backup tapes. This became increasingly annoying as the database grew in size.

As prices dropped on 386 hardware we began using Prime 386/XL UNIX computers for the host. These provided a more traditional implementation of UNIX including a working implementation of the TCP/IP protocol suite, something that was seriously lacking in the version of SCO that we were using. This, coupled with the decision to utilize 286 based PC's as workstations, allowed us to radically rethink the design of the system.

As stated before the initial experiments with Windows 2.0 left a lot to be desired. We chose Digitalk Smalltalk as the graphic front end for the system. It's strengths at the time included the fact that it provided amazing performance on the 286 PC's running plain MSDOS. Smalltalk also allowed for rapid development in an interactive visual environment that is only beginning to be equaled by products such as Visual Basic and the like.

Next we evaluated the various database vendors and chose Oracle, mostly for it's ability to run on the widest variety of platforms. They also seemed to tell the fewest untruths of all the vendors. The particular version of Oracle which we began development with was release 5.0. While it lacked certain features that later would become desirable, it made up for this in robustness and the ability to recover from some fairly spectacular disasters.

Oracle and Smalltalk thus form the basis of the current system. The Clinical Workstations communicate to the host using the UDP portion of the TCP/IP suite. By using this connectionless method of communications we maintain the precept of one blocking input stream into a single reader process. By using UDP we also make the interaction between the host and workstation essentially stateless. This allows us to bring either side (host or workstation) up or down with minimal effect on the other. This is a desirable trait in an environment such as the ED where many people use the system to make short discrete transactions. We also have to face the fact that the PCs are essentially unreliable in the sense that they are subject to the vagaries of both the hostile environment and personnel unfamiliar with computers. In addition, we recognized that the entire system would be left unadministered by the hospital computer services department. With this in mind we tried to make the system run as if it would be left alone in a closet (which indeed several of them are).

This connectionless, stateless communication method also allows the system to be highly scaleable, another desirable trait. As more workstations are added, very little overhead is added since many workstations are in a quiescent state, running only an idle mode "chart rack" representation of the department.

The use of this simple transmission method has allowed us to implement a simple packet-based message protocol with a very low processing overhead for both the host and workstation.

The design of the processes on the host evolved to also contribute to the efficiency of the system. Indeed the original design, as discussed earlier, was for the system to run on a small 386 host. By dividing the processes into functional units and extensively utilizing UNIX message queues for inter-process communication, we are able to maximize the number of transactions handled by the system. Indeed the system consistently handles seven to eight hundred interactions per hour. For example, we handle all communications to the workstations with two processes, a reader deamon called EMREADD and a writer process called EMWRITED.

EMREADD receives a packet from a workstation and pauses only long enough to determine the

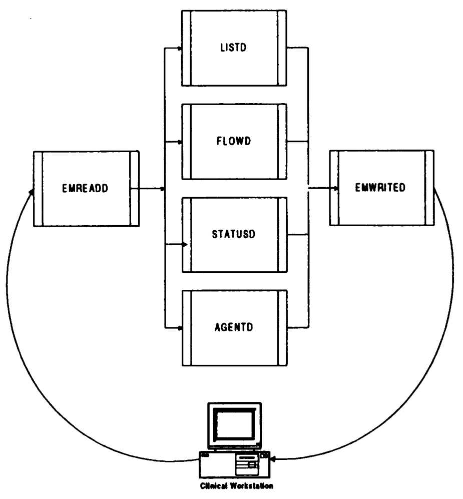

Figure 1 Process Diagram

packet type. These classified packets are then rapidly posted to the appropriate message queue for processing. READD can then block on the input socket and wait for the next packet. While it is in this blocked state it consumes very little in the way of resources, unlike the polling methods used in previous implementations.

All the other deamons can then block on their input queues and wait for something to do.
Such tasks include list requests for fill in the blank forms sent to a list processor called LISTD. Requests for the chart-rack (departmental status) are sent to the status caching process called STATUSD. Requests for specific patient chart information are sent to the flow caching process called FLOWD. All requests to write data to the database are handled through a central broker process called the AGENTD.

An additional benefit to this usage of functional processes is that we have been able to carefully consider and eliminate extremely troublesome race conditions and deadlocks caused by uncontrolled access to the database. Since all queries are processed by the host instead of the workstations, a minimal amount of network traffic is generated and each query can be completely optimized for speed.

After each request has been handled by the appropriate deamon, it is packetized and placed into the message queue for EMWRITED. A header tells EMWRITED whether this packet is destined for a single workstation or if it is to be a multicast to many different stations interested in this particular transaction. The writer is then responsible for handling the transmission and confirming receipt of the packets.

The database has evolved back to the concept of having both a real-time and archive portion. Up to this point Oracle has proven able to manage the real-time and historic data in one set of tables. However as the number of observations in a single system approach the tens of millions, there is a noticeable lag in the amount of time required to bring up current information. If a query degenerated into a sub-optimal form and caused a full table scan to be performed, then it could take ten or fifteen minutes for the query to return. This would certainly degrade the performance of a system where actions are expected to occur in the sub-second time frame.

We were able to break the database back up due to some advances in Oracle's technology. These advances include the UNION ALL construct. UNION ALL allows a query to return all of the rows from multiple tables in a form that resembles a single virtual table. Another function provided is the ability to attach triggers to individual database row actions. Not only can we now manage the data more efficiently by selectively and efficiently replicating it between tables, but we can provide some unique highavailability functions to our users at little or no cost to them.

In summary, the EmStat Emergency Department Clinical Information System has evolved through a series of designs and implementations, each one necessitated by observed practical and organizational difficulties. The current system, while not perfect, is a dramatic improvement over the prototype first conceived in 1986. As in any dynamic system the next stage of evolution is already underway.

References

Galewsky Andrew, Hargrove, James, Owen, Charles B., MD, The EmStat System Design, Clinical Resource Systems, Inc., unpublished, 1986.

Stevens, W. Richard, UNIX Network Programming, Prentice Hall, Englewood Cliffs, NJ, 1990.

Rochkind, Marc J., Advanced UNIX Programming, Prentice Hall, Englewood Cliffs, NJ, 1985.

Tannenbaum, Andrew S., Operating Systems: Design and Implementation, Prentice Hall, Englewood Cliffs, NJ, 1987.

Papadimitrou, Christos, The Theory of Database Concurrency Control, Computer Science Press, Rockville, MD, 1986.

Oracle Corporation, Oracle7 Server Concepts Manual, Oracle Corp., Redwood City, CA., 1992.